Available online at www.sciencedirect.com
ScienceDirect
BioSystems 91 (2008) 545–557
BioSystems
www.elsevier.com/locate/biosystems

Predator–prey dynamics in P systems ruled by metabolic algorithm

F. Fontana *, V. Manca

University of Verona, Department of Computer Science, 15 Strada le Grazie, Verona 37134, Italy

Received 29 March 2006; received in revised form 24 October 2006; accepted 12 December 2006

Abstract

P systems are used to compute predator–prey dynamics expressed in the traditional formulation by Lotka and Volterra. By governing the action of the transition rules in such systems using the regulatory features of the metabolic algorithm we come up with simulations of the Lotka–Volterra equations, whose robustness is comparable to that obtained using Runge-Kutta schemes and Gillespie's Stochastic Simulation Algorithm. Besides their reliability, the results obtained using the metabolic algorithm on top of P systems have a clear biochemical interpretation concerning the role, of *reactants* or *promoters*, of the species involved.

Keywords: Predator–prey systems; Lotka–Volterra dynamics; P systems; Metabolic algorithm

1. Introduction

As for every paradigm belonging to natural computing, membrane systems (also known as P systems) put computation at the heart of the representation of biochemical processes and dynamics (Păun, 2000; Ciobanu et al., 2006). By making use of these systems, and by properly regulating their evolutionary rules via the metabolic algorithm (Manca et al., 2005), in this paper we show that simple membrane models can represent a Lotka–Volterra dynamics once we assign to its species the role of promoting a reaction or taking part into it. By means of these models we will match the performance of computational procedures that are known to simulate the Lotka–Volterra equations with high accuracy both in the deterministic and stochastic regime. In the meantime the membrane model-based simulations preserve a clear biochemical meaning of the computation.

The Lotka–Volterra population dynamics has a long history in natural sciences. Lotka (1920) proposed a set of coupled autocatalytic reactions causing the concentrations of reactants to oscillate:

$$\begin{aligned} &A + X \xrightarrow{c_1} 2X \\ &X + Y \xrightarrow{k_2} 2Y \\ &Y \xrightarrow{k_3} B. \end{aligned} \tag{1}$$

These reactions involve the autocatalysis of a reactant X on the (supposed constantly available) substance A, along with the autocatalysis of a second reactant Y depending on the available concentration of X. The degradation of Y is finally accounted for by a third reaction in which B acts as a sort of "sink" product. Rate coefficients c_1, k_2, and k_3 are assigned to each of these reactions, respectively.

The dynamics of X and Y is described by the following system of ordinary differential equations (Atkins and de

* Corresponding author. Tel.: +39 045 802 7032; fax: +39 045 802 7068.
E-mail addresses: federico.fontana@univr.it (F. Fontana), vincenzo.manca@univr.it (V. Manca).

0303-2647/$ – see front matter © 2007 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2006.12.010

Paula, 2002):

$$x'(t) = k_1 x(t) - k_2 x(t) y(t) \quad (2a)$$

$$y'(t) = k_2 x(t) y(t) - k_3 y(t). \quad (2b)$$

in which it is $k_1 = c_1[A]$ due to the constancy of the concentration of A. Six years after Lotka's research, Volterra presented this differential equation system while independently working on the predator–prey model of a biological population of fish (Volterra, 1926). Since (1) and (2) describe the same dynamics in different contexts, both systems are commonly known under the unified name of Lotka–Volterra reactions, or equations.

Further research has demonstrated that the Lotka–Volterra model has inherent limits as a tool for the mathematical description of both chemical and biological dynamics. About the limits of use in chemistry, it has been shown that the biochemical mechanism underlying the reaction set (1) cannot be reproduced experimentally, and that more sophisticate chemical equations must come into play to realize even a simple reaction exhibiting oscillation (Epstein and Showalter, 1996). About the limits of use in biology, it was soon observed that the system (2) in practice cannot describe a temporal dynamics involving real species, for a Lotka–Volterra system cannot handle even small perturbations affecting an ecosystem (Illner et al., 2005).

Despite this, the Lotka–Volterra equations disclose an interesting family of solutions. It is this trade-off between system simplicity and complexity and solutions richness that has fertilized much research upon them, and makes the Lotka–Volterra model appealing still today.

The system (2) does not yield an analytic solution, but for small deviations from the equilibrium point (Illner et al., 2005). Furthermore, any numerical solution must deal with the *structural instability* affecting the Lotka–Volterra dynamics: due to this instability, even small perturbations injected into the system, or equivalent numerical errors due to inaccuracies of the integration method, cause irreversible deviations of the solution *trajectory* from the unperturbed dynamics (Hilborn, 2000). This implies that non-trivial numerical methods must be used to find out the correct solution of (2) once proper initial conditions for x and y are set.

Explicit Runge-Kutta methods (Iserles, 1996) have demonstrated capable of providing accurate solutions at reasonable computational cost. Conversely, a discretization of (2) made using simple Euler methods results in asymptotically divergent numerical simulations due to structural instability.

The scenario presented so far becomes even richer if we return back to the chemical formulation (1) of the Lotka–Volterra system. As opposed to (2), for this set of reactions a general agreement exists in accepting the asymptotic divergence of the amounts of X and Y as long as *molecules* are considered instead of *concentrations*. Such an agreement has a strong foundation in the solution to (1) provided by the Stochastic Simulation Algorithm, proposed by Gillespie (1976).

Let the reader note that a major difference exists between molecules and concentrations: the former are discrete bodies, hence define positive integer quantities, whereas the latter are continuous by definition and, for this reason, define amounts that are positive real numbers. In the case of Lotka–Volterra this difference makes the two systems somehow even more intertwined one with the other: perhaps curiously, Volterra proposed a continuous system to obtain temporal evolutions of (discrete) populations, while, conversely, the Lotka chemical reactions have been analyzed in detail by Gillespie in the discrete domain.

In conclusion, a dichotomy lies open between the (both acceptable) solutions of the chemical reacting system and its differential equation model:

- if we choose to solve the former *exactly* (in a stochastic sense, and assuming to work with molecule populations instead of concentrations) then we obtain an inaccurate solution of the latter; in the meantime, this solution resembles (in average) the dynamics obtained by applying the simple Euler method to (2);
- conversely, if we decide to solve the latter *exactly* (in a deterministic sense), for instance, using explicit Runge-Kutta methods, then (again in average) we commit an error as compared to solving (1) using Gillespie's algorithm.

We have investigated on this apparent paradox using an alternative method, neither based on Runge-Kutta nor on Gillespie (Bianco et al., 2006b,a). This method results in an algorithm that we call *metabolic* since it has provided promising (although still qualitative) solutions of some biomolecular dynamics such as the leukocyte recruitment, circadian rhythms and the PKC cycle (Fontana et al., 2006; Bianco et al., 2006a). Also, it has already been used to analyze predator–prey dynamics, although in a less systematic way than here (Bianco et al., 2005b).

As we will see in the following, this algorithm models the dynamics of a P system (Păun, 2002) and, for this reason, it works in the *symbolic* domain, i.e., over discrete populations of *symbol/objects*. Despite its general formulation, which includes the presence of multiple membranes and cooperative rules (Bianco et al., 2006b),

we will make limited use of its features, by restricting its definition to non-cooperative rules acting on a single membrane.

In this paper we will model and, hence, solve a Lotka–Volterra dynamics using this method. Similarly to what happens by discretizing this dynamics using Runge-Kutta, the metabolic algorithm finds out a stable solution of the system. It must be stressed here, however, that:

(1) the metabolic algorithm implements a simple and biologically founded *partition* of the population in the system; this simple partitioning is obtained by importing on it some basic properties of P systems;
(2) in order to simulate a Lotka–Volterra system the metabolic algorithm is initialized with a straightforward, self-explaining set of rules which have a clear biochemical interpretation.

The above two points in other words tell that the algorithm does not resort to numerical recipes for making the system evolve, nor it needs to be informed with the structure of an ordinary differential equation system. Despite this it comes up with a dynamics, whose numerical properties are comparable to those exhibited by the explicit Runge-Kutta solution of (2).

In conclusion, the solution of a Lotka–Volterra dynamics provided by a P system ruled by the metabolic algorithm has a clear biochemical interpretation meanwhile preserving good stability properties. This fact contributes to increase our confidence in the metabolic algorithm and, in the end, in P systems as effective tools for the representation of biomolecular phenomena.

We will present our results after providing some insight on the traditional solutions of the Lotka–Volterra system obtained using simple Euler, Runge-Kutta, and Gillespie, as well as after briefly recalling the functioning of the metabolic algorithm.

2. Solution by Numerical Schemes

A comprehensive analysis of the Lotka–Volterra dynamics is best carried out if we inspect the system (2) using reliable numerical tools for this system. This is what we do in this section. Later in the paper we will compare the results obtained here with those coming out by using P systems.

Eqs. (2) give rise to an autonomous system having the following form:

$$x' = g_1(x, y) \tag{3a}$$

$$y' = g_2(x, y). \tag{3b}$$

Here, $x(t)$ and $y(t)$ are functions of the time t.

A numerical solution of (3) can be computed over a uniform subset of the continuous temporal domain. This subset is obtained by substituting the temporal variable t with a discrete set of temporal slots separated by a constant time T: $0, T, 2T, \ldots, nT, \ldots$. Conventionally, $t = 0$ is the initial observation step.

A simple Euler method-based discretization of (3) is obtained by substituting every first derivative with the corresponding discrete (forward) difference (Iserles, 1996) with step T. To explain it better, we will approximate x' over nT by using the formula $x'(nT) \approx \{x(nT + T) - x(nT)\}/T$. It follows that, using the simple Euler method, (3) is approximated by the following numerical system:

$$x(nT + T) = x(nT) + Tg_1(x(nT), y(nT)) \tag{4a}$$

$$y(nT + T) = y(nT) + Tg_2(x(nT), y(nT)). \tag{4b}$$

This corresponds to approximate the solution of the system at step $nT + T$ with a value lying on the tangent to the previous solution point, $x(nT)$, $y(nT)$—note that (3) yields the coefficients of the tangent to any solution point.

An explicit Runge-Kutta scheme of the second order that has a wide field of application considers a midpoint tangent to $x(nT)$, $y(nT)$, i.e., the point at coordinates

$$x_{\mathrm{M}} = x(nT) + \frac{T}{2}x'(nT) \tag{5a}$$

$$y_{\mathrm{M}} = y(nT) + \frac{T}{2}y'(nT). \tag{5b}$$

This point is used as argument in g_1 and g_2 so as to compute the tangent itself more precisely. According to this scheme the numerical solution is given by the following system:

$$x(nT + T) = x(nT) + T\, g_1(x_{\mathrm{M}}, y_{\mathrm{M}}) \tag{6a}$$

$$y(nT + T) = y(nT) + T\, g_2(x_{\mathrm{M}}, y_{\mathrm{M}}). \tag{6b}$$

The application of the schemes (4) and (6) results in the two dynamics expressed by the phase diagrams of Fig. 1. Such dynamics have been plotted for three different initial values of the system (that can be read from the figure) using the parameters $k_1 = k_3 = 3 \times 10^{-2}$ and $k_2 = 4 \times 10^{-5}$, along with a temporal step $T = 0.2$.

The meaning of these dynamics becomes clear if we recall that (2) has been demonstrated to yield *periodic* solutions (Illner et al., 2005). Such solutions result, in a phase diagram, in closed trajectories winding counterclockwise around the fixed point of the system, located

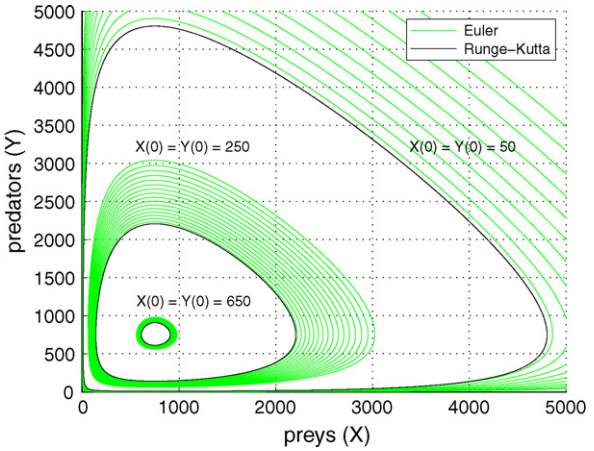

Fig. 1. Numerical solutions ($T = 0.2$) of the Lotka–Volterra system (2) ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$) for three different initial conditions, respectively, using simple Euler method (grey plots) and Runge-Kutta method (black plots).

at coordinates

$$(x_F, y_F) = \left(\frac{k_1}{k_2}, \frac{k_3}{k_2}\right) = (750, 750). \quad (7)$$

From the theory of autonomous systems we know that distinct trajectories never intersect. Hence, we argue that if the iterated map computed by a numerical scheme such as (4) or (6) jumps, step by step, over distinct trajectories of (3), then the numerical solution will be affected by discretization error.

Every new value computed using the simple Euler method lies, by definition, on a straight line tangent to the previous point with coefficients $x'(nT)$, $y'(nT)$—this situation is also depicted in Fig. 2. This implies that almost every computed new point jumps onto a trajectory that is distinct form the previous one.

As a consequence, the simple Euler method results in spiral trajectories progressively moving away from the exact solution. This phenomenon is evident in Fig. 1 (grey plots) in which, for instance, the phase diagrams

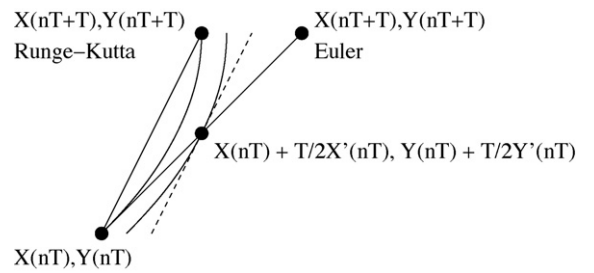

Fig. 2. Comparison between Euler and Runge-Kutta discretization procedures.

trace about 11 oscillation periods in the case with initial conditions $x(0) = y(0) = 50$.

As opposed to Euler, any new discrete point obtained by the Runge-Kutta scheme lies upon a midpoint-corrected tangent (again see Fig. 2). This new point falls exactly over the same trajectory, as the phase diagrams in Fig. 1 confirm. As a result, this scheme is error-free.

Even more accurate are fourth order explicit Runge-Kutta schemes (Iserles, 1996). However, we have checked that the phase diagrams coming out by using fourth order Runge-Kutta nearly superimpose to those obtained using the simpler, second order scheme. Hence, we will consider the plots in Fig. 1 as references for the following of the paper.

3. Solution by Stochastic Simulation

As we have mentioned in Section 1, the Stochastic Simulation (also known as Gillespie's) Algorithm has been used to simulate (1) (Gillespie, 1977). This algorithm provides an exact evolution of a Lotka–Volterra system that is populated by a finite number of individuals.

The term 'exact' has to be intended in a stochastic sense. In the case of Lotka–Volterra, two breakdowns of the oscillatory behavior are possible depending on the species disappearing first: by (1), we argue that if preys disappear first then predators starve to death exponentially; conversely, if predators disappear first then preys reproduce indefinitely, again exponentially.

Independently of how a particular realization terminates, the temporal evolution of a Lotka–Volterra dynamics made using Gillespie exhibits an irregular series of periodical oscillations, whose average amplitude increases until they become so large that one periodic minimum (of either preys or predators) goes to zero. From that point the system starts evolving according to one of the two possible situations seen above.

The Stochastic Simulation Algorithm gives an important lesson: the structural instability of the Lotka–Volterra dynamics cannot be avoided while working with populations. Conversely, concentrations allow for a periodic solution. We will have to deal with this fact while implementing our method in the discrete domain.

4. Solution Using the Metabolic Algorithm

The metabolic algorithm has been presented in previous papers (Manca et al., 2005; Bianco et al., 2006b). Its symbolic nature allows it to work with *evolutionary rules* instead of chemical reactions, furthermore it gener-

alizes molecular (or whatever kind of) populations into *multisets* of objects (Păun and Rozenberg, 2002). Such a formal rearrangement descends from the fact that the metabolic algorithm is framed in the theory of P systems (Păun, 2002), nevertheless the basic working principles of the algorithm can be caught also by the reader who is not familiar with formal language theory (Rozenberg and Salomaa, 1997).

Here we will recall some basic aspects of both P systems and the metabolic algorithm, to make the paper self-contained. Further details can be found in the cited references, particularly about the use of the algorithm with cooperative rules and multiple communicating reaction environments.

The algorithm basic idea is that of partitioning populations of objects at every transition of the system, and then to send these partitions to the evolutionary rules which transform populations into new sets of objects. Every object belongs to a species that is labeled by a corresponding symbol.

The size of each partition depends on the rate coefficients and on the numbers of objects of all species populating the system. These numbers, which vary during the evolution, are contained in the *state* of the system and inform the so-called *reaction maps* (one per rule), whose resulting values are then weighted together in order to find the relative "strength" of every map. The resulting weights are finally used to determine the size of each partition.

In a more general formulation of the metabolic algorithm, the object partitions are assembled together to form strings that are sent, in one shot, to the rules (Bianco et al., 2005a). For the scope of the predator–prey system we will avoid details about the way the metabolic algorithm can process strings, and restrict to the following simplified formalization.

Let us consider a construct $\Pi = (\mathcal{A}, \mathcal{S}, R, F, K)$ in which:

- $\mathcal{A} = \{a_1, \ldots, a_M\}$ is a set of M symbols;
- $\mathcal{S}$ is the set of states, where any state $S \in \mathcal{S}$ gives the cardinality S_a of every symbol a in the system (the initial state gives the cardinality of every symbol in the system in its initial configuration);
- $R = \{r_1, \ldots, r_L\}$ contains L evolutionary rules. Let each rule r_j, $j = 1, \ldots, L$, be in the form $r_j : a_j \to \beta_j$, in which $a_j \in A$ and β_j is a string in $\mathcal{A}^*$, i.e., the set of all possible strings made over the alphabet: $\beta_j \in \mathcal{A}^*$;
- $F = \{f_1, \ldots, f_L\}$ is a set of L functions, called reaction maps, respectively, associated to every rule;
- K is a set of positive real numbers, which occur in the definitions of the reaction maps.

All reaction maps are functions of only K and S. In this way their values are computed from the rate coefficients and the number of objects in the system.

To find the relative strength of a map, we weight its value over the sum of the map values associated to rules sharing the same reactant a. In other words, for every rule r we compute a *reaction weight* W by dividing $f(K, S)$ by the sum of the values $f_i(K, S)$, such that a reacts in r_i. This is done according to the following equation:

$$W_j = \frac{f_j(K, S)}{\sum_{i:\, a_j = \text{left}(r_i)} f_i(K, S)} \tag{8}$$

in which left(r) gives the symbol/reactant contained in r.

Note that Eq. (8) implies

$$\sum_{i:\, a = \text{left}(r_i)} W_i = 1 \qquad \forall a \in \mathcal{A}. \tag{9}$$

We define $|\beta|_a$ to be the number of occurrences of the symbol a contained in the string β. Now, for every $a \in \mathcal{A}$ the reaction weights are used to compute $N(a)$, the number of objects a produced during a transition of the system:

$$N(a) = \sum_{i=1}^{L} |\beta_i|_a \lfloor W_i S_{\text{left}(r_i)} \rfloor. \tag{10}$$

This formula is perhaps worth a comment. For every rule r, a partition of the pool of reactants for that rule is obtained by multiplying $S_{\text{left}(r)}$ by its respective weight—Eq. (9) makes sure that, at every transition, the system cannot consume more reactants than the available ones. The partition value computed by (8) in general must be rounded into an integer number (for the moment we do not address the question about how to round). The rounded partition is then sent to the rule r, which transforms symbol-objects of type left(r) into an equal number of strings β. The number of occurrences of a in β determines the number of objects a produced by r. By summing over all rules we finally find the number of objects a produced during a transition of the system: this is made using (10).

In the end, we compute the variation of the number of objects of every symbol $a \in \mathcal{A}$ during a transition of the system. This number is equal to

$$\Delta a = N(a) - \sum_{i:\, a = \text{left}(r_i)} \lfloor W_i S_a \rfloor. \tag{11}$$

Similarly to (10), the rounding operation appearing in (11) will be made clear in the following.

Eq. (11) is used to update the state. If our P system starts at $t = 0$ and performs a transition every T s, then we can look at the state as a function of the discrete time and write

$$S_a(nT + T) = \Delta a + S_a(nT) \tag{12}$$

for every $a \in \mathcal{A}$.

It must be clear that time plays no role in the metabolic algorithm computation: in fact, the sequence $S_a(nT)$ does not depend on the value of T. Despite this, in the following we will see how the granularity of the observation of the predator–prey system can be tuned using this algorithm, and how this granularity can be put in relationship with the discrete time.

4.1. Formalization of the predator–prey system

In formalizing the Lotka–Volterra dynamics using the metabolic algorithm, a critical decision is left to the experimenter about the role a substance plays in a reaction: if it acts as a reactant then it becomes part of the evolutionary rule accounting for that reaction; conversely, if it plays a promoting role then it is embedded in the reaction map. We recall that P systems with catalysts and P systems with promoters have been already proposed (Păun, 2000; Sosik, 2003; Sburlan, 2006).

Noticeably, neither (1) nor (2) help in this decision. In fact X and Y play exactly the same role in (1), that is, X catalyzes the transformation $A \to X$, and Y does the same for $X \to Y$. Nor (2) provides further hint. So, according to (1) or (2) we have several feasible options depending on whether we treat X and Y as reactants and/or promoters in our algorithm.

For instance, by treating both of them as reactants then we would come up with two rules, $r : AX \to XX$ and $s : XY \to YY$, along with corresponding maps $f_r = c_1$, and $f_s = k_2$ (Bianco et al., 2005b). Conversely, by treating both of them as promoters then we would come up with two rules, $r : A \to X$ and $s : X \to Y$, along with corresponding maps $f_r = c_1 S_X$, and $f_s = k_2 S_Y$. Finally, in both cases it makes sense to represent the degradation of Y using a rule such as $t : Y \to ()$, and to associate this rule to a map $f_t = k_3$.

By exploiting this degree of freedom, from the reaction set (1) we draw the following conclusions:

(1) X reacts in $A + X \to 2X$, and in $X + Y \to 2Y$;
(2) Y promotes the reaction $X + Y \to 2Y$, and it is degraded by $Y \to B$.

Hence, we define the following rules and corresponding maps:

$$\begin{aligned} &r : X \to XX, && f_r = k_1 \\ &s : X \to Y, && f_s = k_2 S_Y \\ &t : Y \to (), && f_t = k_3, \end{aligned} \tag{13}$$

in which we have imported the rate coefficients of the Lotka reactions directly.

Note that we do not need to write r as $AX \to XX$ with $f_r = c_1$ (remind that $k_1 = c_1[A]$): the metabolic algorithm would handle such a rule by transforming into XX the *minimum* number of objects between A and X, or, equivalently, the largest number of couples AX that can be formed with the available objects (Bianco et al., 2005a). As opposed to that, the constant injection of substance A into the reactor (or ecosystem) ensures that there are always enough objects A in the system, and that these objects never limit the execution of r —and, in fact, their concentration is embedded inside k_1 in the differential equation model. Hence, A can be removed from the rule.

According to our interpretation, the first reaction in (1) is a process in which the transformation of AX into XX does not happen in infinitely short time. In fact a molecule X cannot catalyze this reaction and be transformed into Y by the second reaction at the same time. This is equivalent to say that the objects X must be partitioned in two sets: the former is made of objects that will be consumed by r, the latter consists of objects that will be sent to s. Conversely, Y can both catalyze s and be degraded by t.

Notice that the interpretation we give to the first reaction in (1) contradicts its usual physical meaning, appearing in the differential equation model. In fact, Eq. (2a) tells the opposite, i.e., that the pool x is entirely available both for producing new molecules X by means of autocatalysis, and for being transformed into substance Y by means of the second reaction in (1). We can look beyond this apparent contradiction, by reminding that the differential equation set (2) describes the dynamics of the system at every infinitely short temporal step, whereas the metabolic algorithm describes the same dynamics in terms of subsequent assignments of objects to rules. In this sense, the P system-based approach tells something new about the nature of a predator–prey system, at least as far as we observe it in terms of population – not temporal – dynamics.

From (13) we obtain:

$$W_r = \frac{f_r}{f_r + f_s} = \frac{k_1}{k_1 + k_2 S_Y}$$
$$W_s = \frac{f_s}{f_r + f_s} = \frac{k_2 S_Y}{k_1 + k_2 S_Y} \quad (14)$$
$$W_t = \frac{f_t}{f_t} = 1$$

and, finally:

$$\Delta X = \left\lfloor \frac{k_1}{k_1 + k_2 S_Y} S_X \right\rfloor - \left\lfloor \frac{k_2}{k_1 + k_2 S_Y} S_X S_Y \right\rfloor$$
$$\Delta Y = \left\lfloor \frac{k_2}{k_1 + k_2 S_Y} S_X S_Y \right\rfloor - S_Y, \quad (15)$$

which gives the complete information on the system evolution using the metabolic algorithm.

As we have mentioned in Section 1, the Lotka–Volterra system is structurally unstable. This means that, due to the rounding operation appearing in (15), X and Y assume values that jump over distinct solutions of (2). If we want to check the robustness of our solution then we must allow the system (15) to evolve over real numbers. In the real domain Eq. (11) becomes

$$\Delta a = \sum_{i=1}^{L} (|\beta_i|_a - |\text{left}(r_i)|_a) W_i S_{\text{left}(r_i)}. \quad (16)$$

Similarly, the state variation is obtained by removing the rounding operations in (15) in a way that the system evolves according to the following set of equations:

$$x(nT + T) = \Delta x + x(nT) = \frac{2k_1}{k_1 + k_2 y(nT)} x(nT) \quad (17a)$$

$$y(nT + T) = \Delta y + y(nT) = \frac{k_2 y(nT)}{k_1 + k_2 y(nT)} x(nT) \quad (17b)$$

in which we have deliberately denoted the state of the system using lower-case variables, such those seen in (4) and (6), to specify that it is made of real numbers.

The simulation of this predator–prey system by means of the metabolic algorithm gives the phase diagrams in Fig. 3, that have been obtained using the same parameters and initial conditions as those for Euler and Runge-Kutta.

Even a superficial comparison with Fig. 1 reveals major differences between the solution obtained using the metabolic algorithm and those obtained using traditional schemes, particularly with initial condition $x(0) = y(0) = 50$. The evolution obtained using the metabolic algorithm, then, asks for a deeper inspection of its dynamics.

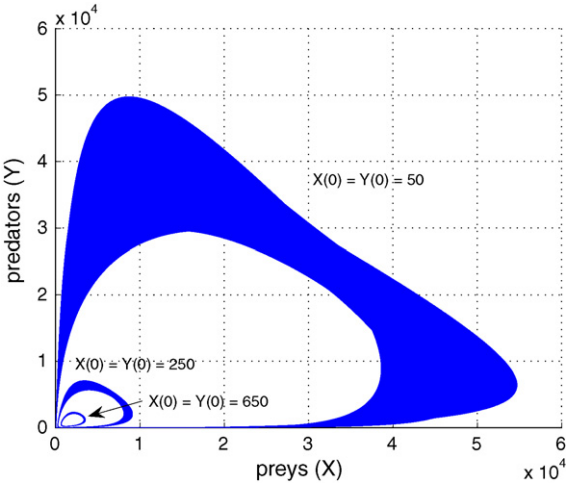

Fig. 3. Numerical solutions of the Lotka–Volterra system ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$) for three different initial conditions, using the metabolic algorithm without rounding (15).

Fig. 4 shows, respectively, in black and grey, plots of the number of predators and preys along 1000 transitions of the P system. These plots disclose that the evolution visible from the phase diagrams is the macroscopic result of a micro-evolution made of continuous jumps to zero followed by climbs to large values. As we will see in the next paragraphs, such an evolution depends on the *granularity of the observation* chosen for the simulation, and can be manipulated by varying this granularity.

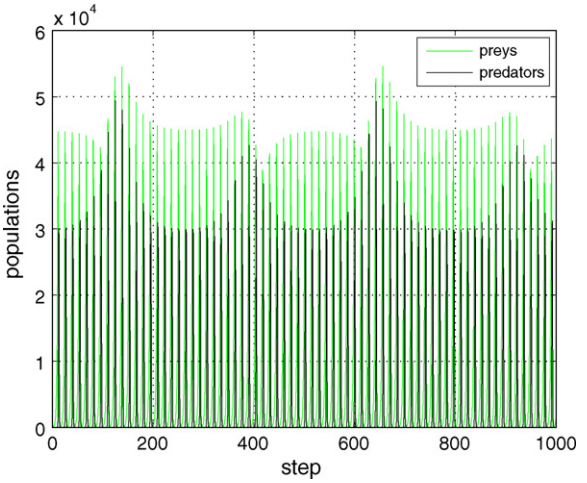

Fig. 4. Number of predators (black plot) and preys (grey plot) along 1000 transitions of the P system, without rounding ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$, $x(0) = y(0) = 50$).

4.2. Granularity of the observation

The rule set (13), along with the weights (14), implies that *all* individuals of both species must take part in the reaction at every transition of the system. In particular, $W_t = 1$ tells that all predators die "in one shot" independently of the value taken by f_t. This means that our observation of the system is coarse-grained: in practice we trigger the P system when all objects have been assigned to a rule.

What happens if we refine our inspection of the system? Must we expect an effect similar to that happening when, in a numerical scheme, we reduce the temporal step (i.e., the value of T), or something else?

A change in the granularity of the observation is straightforward to implement in the metabolic algorithm. The idea behind is to reduce the size of the "active" partitions, by including in the partition set specific classes containing objects that are transformed into themselves.

Let us consider an *inertial rule* (Bianco et al., 2005a), i.e., a rule in the form $u : a \to a$ with $a \in \mathcal{A}$. As opposed to a classic evolutionary rule, due to its structure an inertial rule cannot vary the number and type of objects in the system. In spite of this, by (8) it competes with other rules having form $a \to \beta$: the larger the value of f_u during a transition of the system, the larger the number of objects a "frozen" by u at the same transition. In the limit $f_u = \infty$, all such objects are cut off from the system evolution.

Interestingly, we could avoid the introduction of inertial rules by choosing to build our predator–prey model upon P systems with linked transport, provided with an additional non-reactive (i.e., rule-free) membrane aside of the membrane where the reaction takes place (Păun and Rozenberg, 2002). In this case it would be sufficient to put in the non-reactive membrane a sufficiently large amount of objects X and Y, and provide the reactive membrane with two antiport rules such as (X, out; X, in) and (Y,out;Y,in) in place of the inertial rules, having the same reaction maps as those, respectively, used for the inertial rules. The two antiport rules would in fact send the "frozen" objects to a non-reactive environment, meanwhile recalling objects of the same species back to the membrane where the next reaction step is going to take place. Concerning how large the amount of objects in the non-reacting membrane should be, a study aimed at considering the question in detail should be attempted since, as it can be seen by the first rule in (13), the production of objects X in the reactive membrane admits cases in which the number of preys increases unlimitedly.

Other classes of P systems might probably serve the purpose of inertia in more or less the same way as P systems with linked transport would do. On the other hand, the use of inertial rules makes perhaps the system dynamics easier to be visualized as far as a single reactor is used. Hence, we will stick to inertial rules from here on while considering to express inertia in the context of P systems with linked transport in a future work.

We can change the granularity of the observation of the predator–prey system by including the following inertial rules in (13):

$$\begin{aligned} &u : X \to X \quad f_u \text{ constant} \\ &v : Y \to Y \quad f_v \text{ constant.} \end{aligned} \tag{18}$$

Now, if we push f_u and f_v jointly toward infinity, then we have to expect that the variation in the number of predators and preys over a system transition becomes infinitely small. The following proposition establishes a relationship between this variation and the limit for infinitely small time of the incremental ratios $\Delta x/\Delta t$ and $\Delta y/\Delta t$ that give origin to the system (2).

Proposition 1. *The dynamics of a P system running the metabolic algorithm, provided with rules and reaction maps as in* (13) *and* (18)*, asymptotically converges to the solution of the ODE system* (2) *as soon as* $f_u = f_v \to \infty$.

Proof. Let us extend the rule set (13) by adding u and v as in (18). Then, (14) generalizes into the following set of weights:

$$\begin{aligned} W_r &= \frac{f_r}{f_r + f_s + f_u} = \frac{k_1}{k_1 + k_2 S_Y + f_u} \\ W_s &= \frac{f_s}{f_r + f_s + f_u} = \frac{k_2 S_Y}{k_1 + k_2 S_Y + f_u} \\ W_t &= \frac{f_t}{f_t + f_v} = \frac{k_3}{k_3 + f_v}. \end{aligned} \tag{19}$$

Accordingly, (15) becomes

$$\begin{aligned} \Delta X &= \left\lfloor \frac{k_1}{k_1 + k_2 S_Y + f_u} S_X \right\rfloor - \left\lfloor \frac{k_2}{k_1 + k_2 S_Y + f_u} S_X S_Y \right\rfloor \\ \Delta Y &= \left\lfloor \frac{k_2}{k_1 + k_2 S_Y + f_u} S_X S_Y \right\rfloor - \left\lfloor \frac{k_3}{k_3 + f_v} S_Y \right\rfloor. \end{aligned} \tag{20}$$

Hence, (17) can be rewritten in the following way:

$$\begin{aligned} x(nT + T) = x(nT) &+ \frac{k_1}{d(nT)} x(nT) \\ &- \frac{k_2}{d(nT)} x(nT) y(nT) \end{aligned} \tag{21a}$$

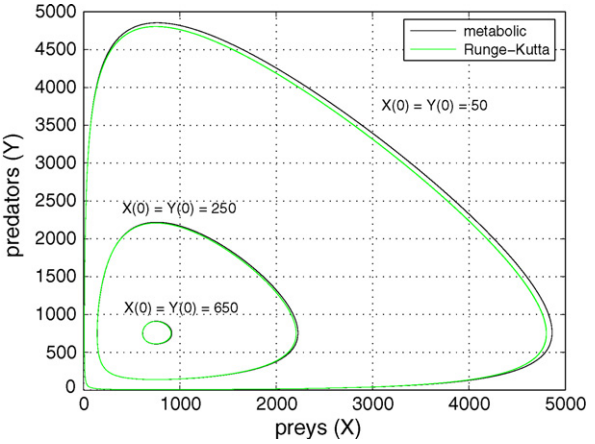

Fig. 5. Phase diagrams of the predator–prey system ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$) for three different initial conditions, using (22) ($f_u = f_v = 5$, black plots) and (6) ($T = 0.2$, grey plots).

$$y(nT + T) = y(nT) + \frac{k_2}{d(nT)} x(nT)y(nT) - \frac{k_3}{k_3 + f_v} y(nT) \tag{21b}$$

with $d(nT) = k_1 + k_2 y(nT) + f_u$.

Eq. (21) establishes a relation between two subsequent transitions of the P system which does not depend on a time reference. In other words, it is true at every time t:

$$x(t + T) = x(t) + \frac{k_1}{d(t)} x(t) - \frac{k_2}{d(t)} x(t)y(t) \tag{22a}$$

$$y(t + T) = y(t) + \frac{k_2}{d(t)} x(t)y(t) - \frac{k_3}{k_3 + f_v} y(t). \tag{22b}$$

For sufficiently large values of f_u and f_v the approximations $d(t) \approx f_u$ and $k_3 + f_v \approx f_v$ can be used inside (22):

$$x(t + T) \approx x(t) + \frac{k_1}{f_u} x(t) - \frac{k_2}{f_u} x(t)y(t) \tag{23a}$$

$$y(t + T) \approx y(t) + \frac{k_2}{f_u} x(t)y(t) - \frac{k_3}{f_v} y(t). \tag{23b}$$

If we equate f_u and f_v by setting $f_u = f_v = 1/T$, then (23) can be rewritten as

$$x(t + T) \approx x(t) + T\{k_1 x(t) - k_2 x(t)y(t)\} \tag{24a}$$

$$y(t + T) \approx y(t) + T\{k_2 x(t)y(t) - k_3 y(t)\}. \tag{24b}$$

Now, if f_u and f_v go to infinity (e.g., the granularity of the observation becomes infinitely fine) the approximations turn into equalities and we get

$$\begin{aligned} \lim_{T \to 0} \frac{x(t + T) - x(t)}{T} &= x'(t) = k_1 x(t) - k_2 x(t)y(t) \\ \lim_{T \to 0} \frac{y(t + T) - y(t)}{T} &= y'(t) = k_2 x(t)y(t) - k_3 y(t). \end{aligned} \tag{25}$$

Hence, (22a) and (22b) solve (2). □

However, we must never forget the difference existing between the concept of granularity of observation in the metabolic algorithm and in traditional numerical schemes: the former splits populations, the latter time.

This difference has practical consequences in the simulations. Firstly, the two approaches yield similar (but never identical!) results only if the following conditions hold:

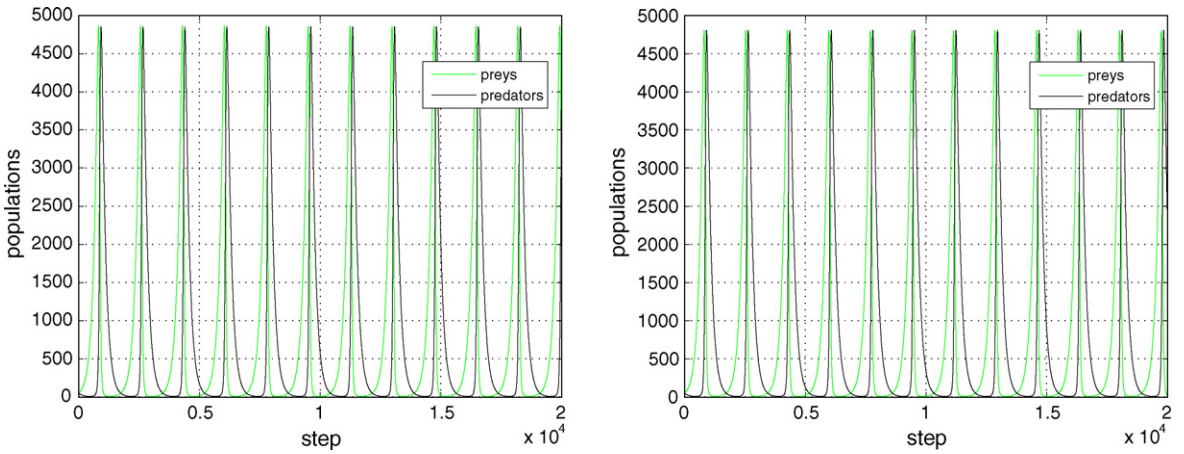

Fig. 6. Evolution of the predator–prey system ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$, $x(0) = y(0) = 50$) obtained using the metabolic algorithm ($f_u = f_v = 5$, left) and Runge-Kutta ($T = 0.2$, right).

- all the inertial rules have identical reaction maps;
- at every transition of the system such maps are large, compared to those being associated to the other (evolutionary) rules;
- every transition of the P system is supposed to span a time lasting $T = 1/f_u = 1/f_v$ s.

For these reasons, a temporal reading of the species evolution in the P system is in theory open to objections (i.e., correct only in the limit of infinitely fine granularity of the observation), and in practice prone to non-negligible simulation error.

Secondly, it is interesting to note that the granularity of the observation has a finite upper bound in correspondence of $f_u = f_v = 0$. As we have seen in Section 4, this choice corresponds to send all objects to non-inertial evolutionary rules at every transition of the system.

As opposed to substance-based granularity, time-based granularity is not upper bounded since T can range in principle toward infinity. However, we must remember that deterministic simulation methods compute discrete molecular dynamics exactly only if the populations are indefinitely large (Gillespie, 2001). Under this assumption, a P system in which inertial rules are absent transforms infinitely many objects at every transition as a numerical scheme would do by setting $T = \infty$.

Eq. (24) also tell that, in practice, comparable granularity can be obtained by setting the values of the inertial rule maps to the same magnitude order as the inverse of the temporal step size. In the simulations using Runge-Kutta, reported in Fig. 1, it was $T = 0.2$.

Fig. 5 shows, in black color, the results obtained by parametrizing the metabolic algorithm with the same values used for Runge-Kutta, and with $f_u = f_v = 1/T = 5$. For ease of comparison the plots obtained with Runge-Kutta are repeated in grey lines. An inspection of these plots confirms that the behavior of the simulations agrees with the conclusions we have drawn based on the observation of the equations.

Subtle differences exist in the amplitudes (this fact is visible in the diagrams of Fig. 5, from which we can see that the metabolic algorithm produces slightly wider amplitudes) and also in the oscillation periods: Fig. 6 in fact shows that the oscillations obtained using the metabolic algorithm starting with initial conditions $x(0) = y(0) = 50$ (left plot) are slightly slower than those obtained with Runge-Kutta, with the same initial condition and using identical parameters (right plot).

5. Evolution with Discrete Populations

So far we have observed two facts:

- the metabolic algorithm leads to a robust solution of the Lotka–Volterra system;
- by refining the granularity of the observation, this solution converges to that obtained with traditional numerical schemes.

We have come up to such conclusions by computing with real numbers. This corresponds to consider fractions of objects if we work in finite-arithmetic precision.

On the other hand, the structural instability of the Lotka–Volterra equations prevents from predicting the system evolution when the state is defined over integer numbers. In fact, if we look at a rounding operation as to a *small* perturbation injected into our system, then by definition of structural instability we know that the

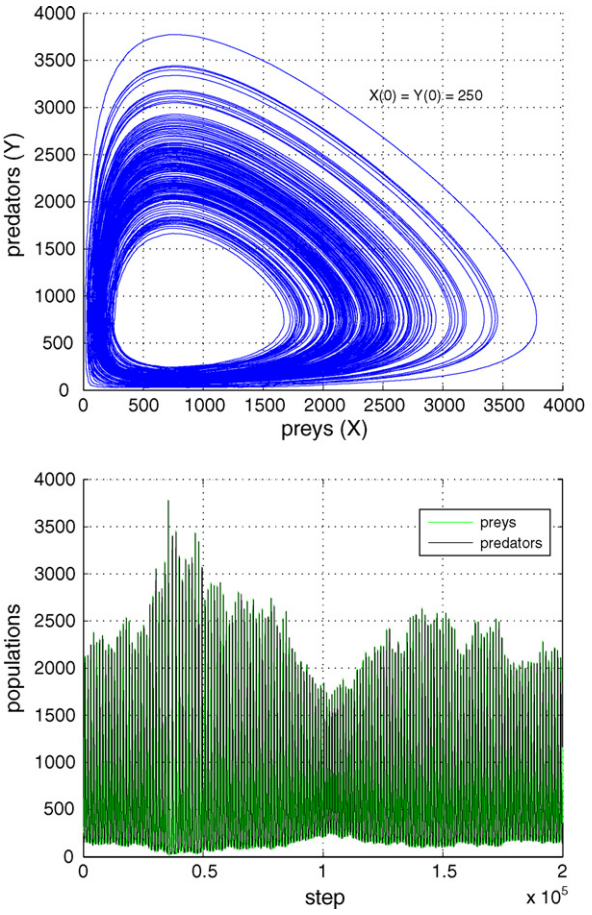

Fig. 7. Phase diagram (above) and evolution (below) of a realization of the predator–prey system using the metabolic algorithm ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$, $f_u = f_v = 5$, $X(0) = Y(0) = 250$).

system in general will not respond to this perturbation with a proportionally *small* deviation of the solution from the original (unperturbed) trajectory (Iserles, 1996). In a few words, the Lotka–Volterra system is not resilient against small perturbations.

In the field of digital signal processing, rounding effects are modeled (if possible) using Gaussian noise sources (Oppenheim and Schafer, 1989). In fact, theorems exist which prove the equivalence of the rounding operation with an injection of Gaussian noise to the system holding the hypothesis of small perturbations (Oppenheim and Schafer, 1989). Unfortunately such theorems are not valid for our system, since situations occur in which few individuals populate it: in this case rounding introduces relatively large perturbations, that cannot be modeled as an additional noise source.

Our rounding procedure tries to imitate the additional noise model in a way that an integer value X resulting from this procedure resembles, as much as it can, the output x of a system perturbed by a Gaussian noise source w. The procedure is made of the following steps:

(1) the value w_x of a Gaussian random process (Oppenheim and Schafer, 1989) with null mean and standard deviation equal to 0.25 is generated;
(2) if $|w_x| > 0.5$ then the magnitude of w_x is truncated to 0.5;
(3) the integer value X is obtained by rounding the sum $x + w_x$ to the nearest integer: $X = \lfloor x + w_x \rceil$.

In this way the evolution of the P system is non-deterministic, meanwhile at every transition the number of objects populating the system results by rounding a real number which includes the presence of Gaussian noise.

Fig. 7 shows simulation results, both in terms of phase diagrams (left plot) and species evolution (right plot), of

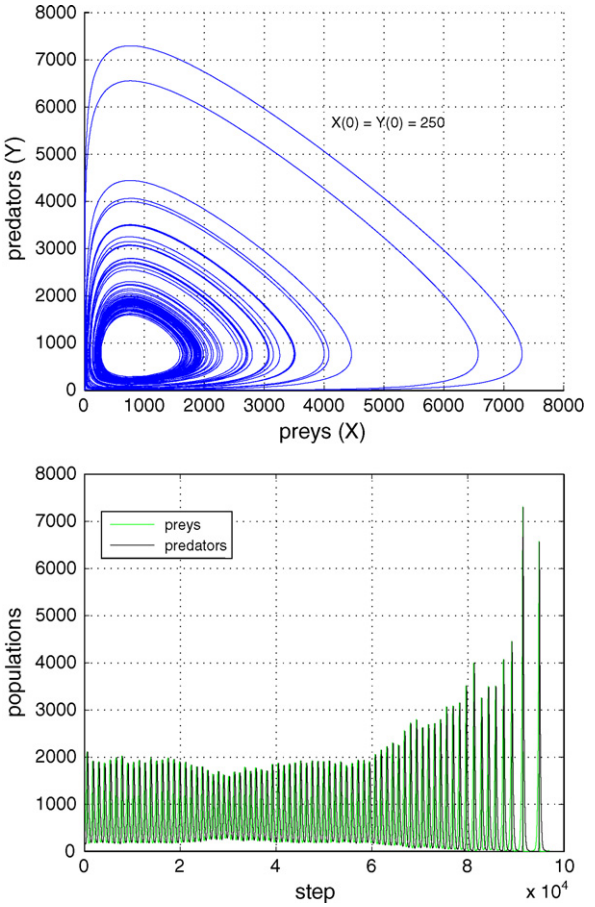

Fig. 8. Phase diagram (above) and evolution (below) of an unstable realization of the predator–prey system using the metabolic algorithm ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$, $f_u = f_v = 5$, $X(0) = Y(0) = 250$).

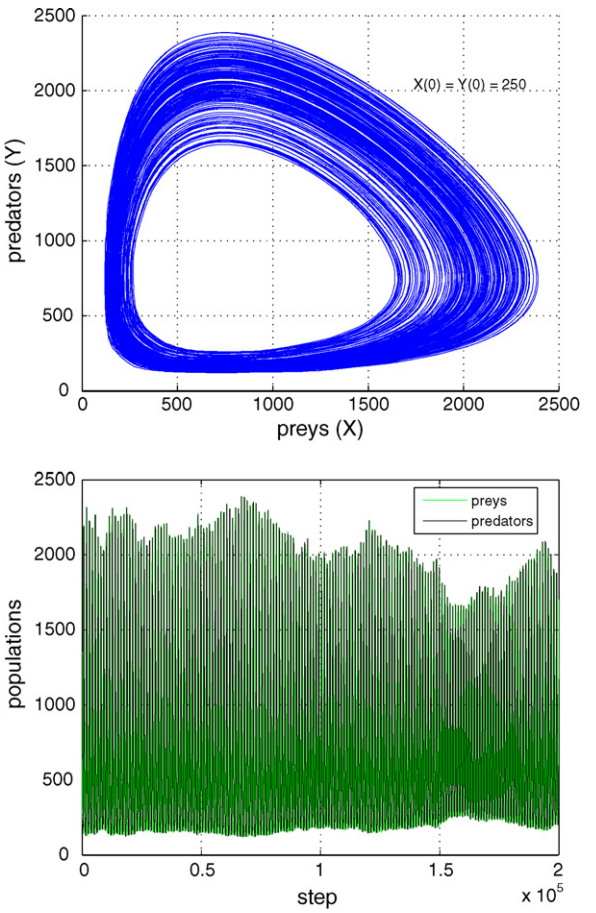

Fig. 9. Phase diagram (above) and evolution (below) of a realization of the predator–prey system using Runge-Kutta ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$, $f_u = f_v = 5$, $X(0) = Y(0) = 250$).

a realization of the Lotka–Volterra system based on the following modified version of (20):

$$\Delta X = \left\lfloor \frac{k_1 S_X}{k_1 + k_2 S_Y + f_u} + w_r \right\rfloor - \left\lfloor \frac{k_2 S_X S_Y}{k_1 + k_2 S_Y + f_u} + w_s \right\rfloor,$$

$$\Delta Y = \left\lfloor \frac{k_2 S_X S_Y}{k_1 + k_2 S_Y + f_u} + w_s \right\rfloor - \left\lfloor \frac{k_3 S_Y}{k_3 + f_v} + w_t \right\rfloor, \quad (26)$$

in which w_r, w_s and w_t are Gaussian random processes.

Preys eventually drop to zero, or diverge. Fig. 8 shows an example in which both species drop to zero after about 10^5 transitions of the system due to the death of preys.

The reader may still wonder whether a rounded version of Runge-Kutta leads to similar results. The answer is positive: by modifying Eqs. (6a) and (6b) in this way:

$$x(nT + T) = x(nT) + \lfloor T\, g_1(x_M, y_M) + w_x \rfloor \quad (27a)$$

$$y(nT + T) = y(nT) + \lfloor T\, g_2(x_M, y_M) + w_y \rfloor, \quad (27b)$$

with the usual meaning of the operator $\lfloor\ \rfloor$, and w_x and w_y being Gaussian random processes, then we come up with the results depicted in Figs. 9 and 10, respectively, in the case when preys survive and die within the observed range, using the same parameters and initial conditions as those employed to produce the plots in Figs. 7 and 8.

At this point, it is not surprising that a simulation based on Gillespie's algorithm leads to results that are similar to those presented in Figs. 7–10 (Gillespie, 1977). These results, in fact, are affected by the same structural instability as those obtained by the metabolic algorithm as well as the rounded Runge-Kutta scheme.

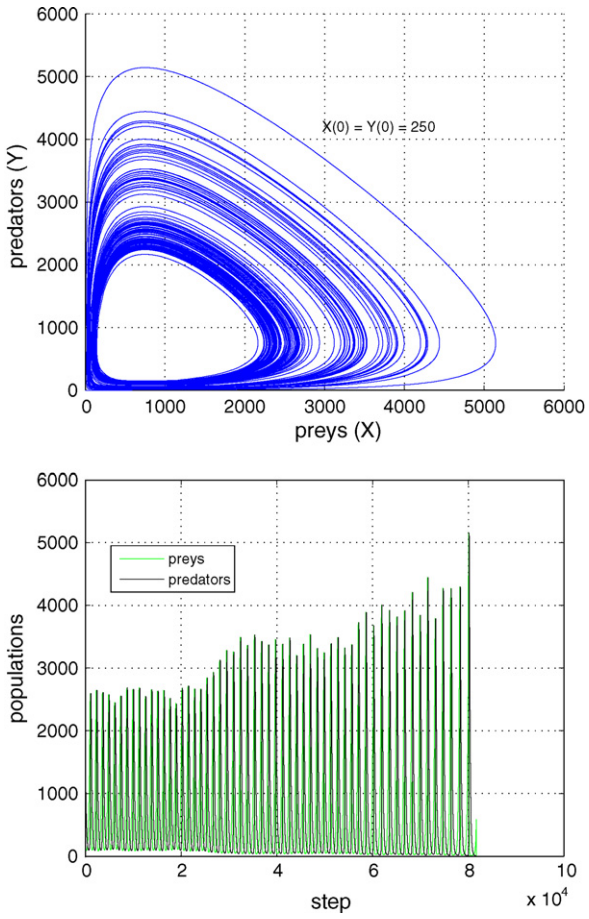

Fig. 10. Phase diagram (above) and evolution (below) of an unstable realization of the predator–prey system using Runge-Kutta ($k_1 = k_3 = 3 \times 10^{-2}$, $k_2 = 4 \times 10^{-5}$, $f_u = f_v = 5$, $X(0) = Y(0) = 250$).

6. Conclusions

In this study we have shown the capability of a P system, evolving according to the rules provided by the metabolic algorithm, to come up with correct and robust simulations of the Lotka–Volterra dynamics.

The rule set used in the P system adds insight on the behavior and nature of the species populating the ecosystem, as far as this dynamics is looked at as a series of subsequent partitions of populations and not as a temporal evolution of the system state.

Correctness has been demonstrated by comparing the results of simulations made with a P system, modified to handle fractions of objects, to figures taken from an explicit Runge-Kutta numerical scheme whose reliability in solving Volterra's differential equations has widely been recognized.

Robustness has been addressed in the inherent stability limits of Lotka's reaction set, by comparing the results obtained with the P system with those found by Gillespie as well as by restricting the Runge-Kutta scheme to process integer numbers.

In order to let the metabolic algorithm work over discrete populations of objects we had to implement a rounding procedure, that can be shown to be equivalent to adding Gaussian noise to the system only when the number of objects in the system is large. Such a procedure will probably become unnecessary in a future version of the algorithm, in which finite populations are expected to be processed instead of finite concentrations, the same way as it happens in stochastic algorithms.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive criticism, in particular for suggesting links between inertial rules and existing classes of P systems as those reported in Section 4.2.

References

Atkins, P., de Paula, J., 2002. Physical Chemistry, seventh ed. Oxford University Press, Oxford, UK.

Bianco, L., Fontana, F., Franco, G., Manca, V., 2006a. P systems for biological dynamics. In: Ciobanu, G., Păun, G., Pérez-Jiménez, M.J. (Eds.), Applications of Membrane Computing. Springer, pp. 81–126.

Bianco, L., Fontana, F., Manca, V., 2005a. Metabolic algorithm with time-varying reaction maps. In: Proceedings of the Third Brainstorming Week on Membrane Computing (BWMC3), Sevilla, Spain, pp. 43–62.

Bianco, L., Fontana, F., Manca, V., 2005b. Reaction-driven membrane systems. In: Wang, L., Chen, K., Ong, Y.-S. (Eds.), Advances in Natural Computation, First International Conference, ICNC 2005, Changsha, China, August 27–29, 2005, Proceedings, Part II, vol. 3611 of Lecture Notes in Computer Science. Springer, pp. 1155–1158.

Bianco, L., Fontana, F., Manca, V., 2006b. P systems with reaction maps. Int. J. Found. Comput. Sci. 17 (1), 27–48.

Ciobanu, G., Păun, G., Pérez-Jiménez, M.J. (Eds.), 2006. Applications of Membrane Computing. Springer, Berlin.

Epstein, I.R., Showalter, K., 1996. Nonlinear chemical dynamics: Oscillations, patterns, and chaos. J. Phys. Chem. 100 (31), 13132–13147.

Fontana, F., Bianco, L., Manca, V., 2006. P systems and the modeling of biochemical oscillations. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (Eds.), Sixth Workshop on Membrane Computing (WMC6), vol. 3850 of Lecture Notes in Computer Science, Springer, pp. 199–208.

Gillespie, D.T., 1976. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comput. Phys. 22, 403.

Gillespie, D.T., 1977. Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81 (25), 2340–2361.

Gillespie, D.T., 2001. Approximate accelerated stochastic simulation of chemically reacting systems. J. Chem. Phys. 115 (4), 1716–1733.

Hilborn, R.C., 2000. Chaos and Nonlinear Dynamics. Oxford University Press, Oxford, UK.

Illner, R., Bohun, C.S., McCollum, S., van Roode, T., 2005. Mathematical Modelling. American Mathematical Society, Providence, RI.

Iserles, A., 1996. Numerical Analysis of Differential Equations. Cambridge University Press, Cambridge, MA.

Lotka, A.J., 1920. Undamped oscillations derived from the law of mass action. J. Am. Chem. Soc. 42, 1595–1599.

Manca, V., Bianco, L., Fontana, F., 2005. Evolutions and oscillations of P systems: Applications to biological phenomena. In: Mauri, G., Păun, G., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (Eds.), Membrane Computing, Fifth International Workshop, vol. 3365 of Lecture Notes in Computer Science. Springer, pp. 63–84.

Oppenheim, A.V., Schafer, R.W., 1989. Discrete-Time Signal Processing. Prentice-Hall, Inc., Englewood Cliffs, NJ.

Păun, G., 2000. Computing with membranes. J. Comput. Syst. Sci. 61 (1), 108–143.

Păun, G., 2002. Membrane Computing. An Introduction. Springer, Berlin.

Păun, G., Rozenberg, G., 2002. A guide to membrane computing. Theor. Comput. Sci. 287, 73–100.

Rozenberg, G., Salomaa, A. (Eds.), 1997. Handbook of Formal Languages. Springer-Verlag, Berlin, Germany.

Sburlan, D., 2006. Further results on P systems with promoters/inhibitors. Int. J. Found. Comput. Sci. 17 (1), 206–222.

Sosik, P., 2003. The power of catalysts and priorities in membrane systems. Grammars 6 (1), 13–24.

Volterra, V., 1926. Fluctuations in the abundance of a species considered mathematically. Nature 118, 558–560.